Using Term to Pierce an Internet Firewall

Table of Contents

Using Term to Pierce an Internet Firewall....1
Barak Pearlmutter, bap@cs.unm.edu....1
1.Disclaimer....1
2.Copyright....1
3.Introduction....1
4.The basic procedure....1
5.Detailed directions....1
6.Multiple term sockets....1
7.The ~/.term/termrc.telnet init file....1
8.Direction....1
9.Security....2
10.Telnet mode....2
11.Bugs and term wish list....2
12.Tricks that don't seem to work....2
13.Related resources....2
14.Acknowledgments....2
1.Disclaimer....2
2.Copyright....2
3.Introduction....3
4.The basic procedure....3
5.Detailed directions....3
6.Multiple term sockets....4
7.The ~/.term/termrc.telnet init file....5
8.Direction....5
9.Security....5
10.Telnet mode....5
11.Bugs and term wish list....6
12.Tricks that don't seem to work....6
13.Related resources....6
14.Acknowledgments....6

Using Term to Pierce an Internet Firewall

Barak Pearlmutter, `bap@cs.unm.edu`

v, 15 July 1996

Directions for using ``term'' to do network stuff through a TCP firewall that you're not supposed to be able to.

1.Disclaimer

2.Copyright

3.Introduction

4.The basic procedure

5.Detailed directions

6.Multiple term sockets

7.The `~/.term/termrc.telnet` init file

8.Direction

9.Security

10.Telnet mode

11.Bugs and term wish list

12.Tricks that don't seem to work

13.Related resources

14.Acknowledgments

1.Disclaimer

!!! READ THIS IMPORTANT SECTION !!!

I hereby disclaim all responsibility for this hack. If it backfires on you in any way whatsoever, that's the breaks. Not my fault. If you don't understand the risks inherent in doing this, don't do it. If you use this hack and it allows vicious hackers to break into your company's computers and costs you your job and your company millions of dollars, well that's just tough nuggies. Don't come crying to me.

2.Copyright

Unless otherwise stated, Linux HOWTO documents are copyrighted by their respective authors. Linux HOWTO documents may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies. Commercial redistribution is allowed and encouraged; however, the author would like to be notified of any such distributions.

All translations, derivative works, or aggregate works incorporating any Linux HOWTO documents must be covered under this copyright notice. That is, you may not produce a derivative work from a HOWTO and impose additional restrictions on its distribution. Exceptions to these rules may be granted under certain conditions; please contact the Linux HOWTO coordinator at the address given below.

In short, we wish to promote dissemination of this information through as many channels as possible. However, we do wish to retain copyright on the HOWTO documents, and would like to be notified of any

plans to redistribute the HOWTOs.

If you have questions, please contact Tim Bynum, the Linux HOWTO coordinator, at linux−howto@sunsite.unc.edu via email.

3.Introduction

The "term" program is normally used over a modem or serial line, to allow various host−to−host services to flow along this simple serial connection. However, sometimes it is useful to establish a term connection between two machines that communicate via telnet. The most interesting instance of this is for connecting two hosts which are separated by ethernet firewalls or SOCKS servers. Such firewalls provides facilities for establishing a telnet connection through the firewall, typically by using the SOCKS protocol to allow inside machines to get connections out, and requiring outside users to telnet first to a gateway machine which requires a one−time password. These firewalls make it impossible to, for instance, have X clients on an inside machine communicate with an X server on an outside machine. But, by setting up a term connection, these restrictions can all be bypassed quite conveniently, at the user level.

4.The basic procedure

Setting up a term connection over a telnet substrate is a two−phase process. First your usual telnet client is used to set up a telnet connection and log in. Next, the telnet client is paused and control of the established telnet connection is given to term.

5.Detailed directions

In detail, the process goes like this.

First, from a machine inside the firewall, telnet to a target machine outside the firewall and log in.

Unless you are under linux and will be using the proc filesystem (see below) make sure your shell is an sh style shell. Ie if your default shell is a csh variant, invoke telnet by

```
(setenv SHELL /bin/sh; telnet machine.outside)
```

After logging in, on the remote (outside) machine invoke the command

```
term −r −n off telnet
```

Now break back to the telnet prompt on the local (inside) machine, using `^]` or whatever, and use the telnet shell escape command `!` to invoke term,

```
telnet> ! term -n on telnet >&3 <&3
```

Et voila!!!

(If you have a variant telnet, you might have to use some other file descriptor than 3; easy to check using strace. But three seems to work on all bsd descendent telnet clients I've tried, under both SunOS 4.x and the usual linux distributions.)

Some telnet clients do not have the ! shell escape command. Eg the telnet client distributed with Slackware 3.0 is one such client. The sources that the Slackware telnet client is supposedly built from,

```
ftp://ftp.cdrom.com:/pub/linux/slackware-3.0/source/n/tcpip/NetKit-B-0.05.tar.gz
```

have the shell escape command. A simple solution is therefore to obtain these sources and recompile them. This unfortunately is a task I have had no luck with. Plus, if you are running from inside a SOCKS firewall, you will need a SOCKSified telnet client anyway. To that end, I was able to compile a SOCKSified telnet client from

```
ftp://ftp.nec.com/pub/security/socks.cstc/socks.cstc.4.2.tar.gz
```

or if you're outside the USA,

```
ftp://ftp.nec.com/pub/security/socks.cstc/export.socks.cstc.4.2.tar.gz
```

Alternatively, under linux kernels up to 1.2.13, you can pause the telnet with `^]^z`, figure out its pid, and invoke

```
term -n on -v /proc/<telnetpid>/fd/3 telnet
```

This doesn't work with newer 1.3.x kernels, which closed some mysterious security hole by preventing access to these fd's by processes other than the owner process and its children.

6.Multiple term sockets

It is a good idea to give the term socket an explicit name. This is the "`telnet`" argument in the invocations of term above. Unless you have the TERMSERVER environment variable set to telnet as appropriate, you invoke term clients with the –t switch, e.g. "`trsh -t telnet`".

7.The `~/.term/termrc.telnet` init file

I have checked line clarity using linecheck over this medium. I expected it to be completely transparent, but it is not. However, the only bad character seems to be 255. The `~/.term/termrc.telnet` I use (the `.telnet` is the name of the term connection, see above) contains:

```
baudrate off
escape 255
ignore 255
timeout 600
```

Perhaps it could be improved by diddling, I am getting a throughput of only about 30k cps over a long−haul connection through a slow firewall. Ftp can move about 100k cps over the same route. A realistic baudrate might avoid some timeouts.

8.Direction

Obviously, if you are starting from outside the firewall and zitching in using a SecureID card or something, you will want to reverse the roles of the remote vs local servers given above. (If you don't understand what this means, perhaps you are not familiar enough with term to use the trick described in this file responsibly.)

9.Security

This is not much more of a vulnerability than the current possibility of having a telnet connection hijacked on an unsecured outside machine. The primary additional risk comes from people being able to use the term socket you set up without you even being aware of it. So be careful out there. (Personally, I do this with an outside machine I know to be pretty secure, namely a linux laptop I maintain myself that does not accept any incoming connections.)

Another possibility is to add "socket off" to the remote `~/.term/termrc.telnet`, or `add "−u off"` to invocation of term. This prevents the socket from being hijacked from the remote end, with only a minor loss of functionality.

10.Telnet mode

Be sure the remote telnetd is not in some nasty seven−bit mode. Or if it is, you have to tell term about it when you invoke term, by adding the `−a` switch at both ends. (I sometimes use "`^] telnet> set outbin`" or "`set bin`" or invoke telnet with a `−8` switch to put the connection into eight−bit mode.)

11.Bugs and term wish list

The linecheck program has some problems checking telnet connections sometimes. This is sometimes because it doesn't check the return code of the `read()` call it makes. For network connections, this call to `read()` can return `-1` with an `EINTR` (interrupted) or `EAGAIN` (try again) error code. Obviously this should be checked for.

There are a number of features that could ease the use of term over telnet. These primarily relate to an assumption that influenced the design of term, namely that the connection is low bandwidth, low latency, and somewhat noisy.

A telnet connection is in general high bandwidth, high latency, and error free. This means that the connection could be better utilized if (a) the maximum window size was raised, well above the limit imposed by term's `N_PACKETS/2=16`, (b) there was an option to turn off sending and checking packet checksums, and (c) larger packets were permitted when appropriate.

Also, to enhance security, it would be nice to have a term option to log all connections through the socket it monitors to a log file, or to stderr, or both. This would allow one to see if one's term connection is being subverted by nasty hackers on the outside insecure machine.

12.Tricks that don't seem to work

Some telnet clients and servers agree to encrypt their communications, to prevent evesdropping on the connection. Unfortunately, the hack used above (using the network connection that the telnet client has set up while the telnet client is idle) won't work in that case. Instead, one really must go through the telnet client itself, so it can do its encryption. It seems like that requires a simple hack to the telnet client itself, to add a command that runs a process with its stdin and stdout are connected to the live telnet connection. This would also be useful for various 'bots, so perhaps someone has already hacked it up.

13.Related resources

A vaguely related trick is to SOCKSify one's Term library. Details, including patches to SOCKS, are available from Steven Danz <danz@wv.mentorg.com>.

14.Acknowledgments

Thanks for valuable suggestions from:

- Gary Flake <flake@scr.siemens.com>
- Bill Riemers <bcr@physics.purdue.edu>
- Greg Louis <glouis@dynamicro.on.ca>

Extra copy of IMPORTANT DISCLAIMER −−− BELIEVE IT!!!

I hereby disclaim all responsibility for this hack. If it backfires on you in any way whatsoever, that's the breaks. Not my fault. If you don't understand the risks inherent in doing this, don't do it. If you use this hack and it allows vicious hackers to break into your company's computers and costs you your job and your company millions of dollars, well that's just tough nuggies. Don't come crying to me.